HARD REAL-TIME, PIXEL-PARALLEL RENDERING OF LIGHT FIELD VIDEOS USING STEERED MIXTURE-OF-EXPERTS

Ignace P. Saenen*, Ruben Verhack*†, Vasileios Avramelos* Glenn Van Wallendael*, and Peter Lambert*
* Ghent University - imec, IDLab, Department of Electronics and Information Systems (ELIS), Belgium
† Technische Universität Berlin, Communication Systems Lab, Germany

***Abstract*—Steered Mixture-of-Experts (SMoE) is a novel framework for the approximation, coding, and description of image modalities such as light field images and video. The future goal is to arrive at a representation for Six Degrees-of-Freedom (6DoF) image data. Previous research has shown the feasibility of real-time pixel-parallel rendering of static light field images. Each pixel is independently reconstructed by kernels that lay in its vicinity. The number of kernels involved forms the bottleneck on the achievable framerate. The goal of this paper is twofold. Firstly, we introduce pixel-level rendering of light field video, as previous work only rendered static content. Secondly, we investigate rendering using a predefined number of most significant kernels. As such, we can deliver hard real-time constraints by trading off the reconstruction quality.**

I. Introduction

Our goal is to achieve the so-called *Six Degrees-of-Freedom* (6DoF) for camera captured images. Currently, 360° video is the main form of virtual reality for camera captured scenes. This does not include any translational freedom towards the viewer, only rotational head-movements are available. Currently MPEG started standardization efforts for a 6DoF format [1]. Their envisioned process consists of two steps: (1) find the most important views on a scene, (2) encode these views using well-known difference- and transform coding approaches. At decoder side, views are synthesized potentially with extra transmitted geometrical side-information. We identify three issues here. Firstly, we argue that 2D regular sampling grids are not optimal representations for storing high-dimensional data. Secondly, we believe that the view synthesis process could shift considerable computational complexity towards the decoder. Finally, note the high variance of the decoding speed of different frame types that exists due to the difference-coding techniques.

The 2D images observed by humans at each angle are processed versions of the higher-dimensional data the camera sensor has acquired. In terms of signal processing, we are likely presented with a high-dimensional sampling problem with nonuniform and nonlinear sample spacing and high-dimensional spatio-directionally varying sampling kernels [2]. The high-dimensional space is defined by the 5D plenoptic function [3]. However, when there are no occlusions (i.e. "open space" assumption), the 5D space can be reduced to the 4D light field [4], [5]. This assumption does not hold for 6DoF in large scenes, however, at the moment this is a widely used simplification [5].

Therefore, we previously introduced a novel methodology that aims to provide full 6DoF, namely *Steered Mixture-of-Experts* (SMoE). We directly model the underlying plenoptic function (or a lower-dimensional projection) in a continuous, analytical form [3]. Currently, we successfully applied SMoE on images, video, 4D light field images and video [6]–[9]. As such we are nearing a full 6DoF representation.

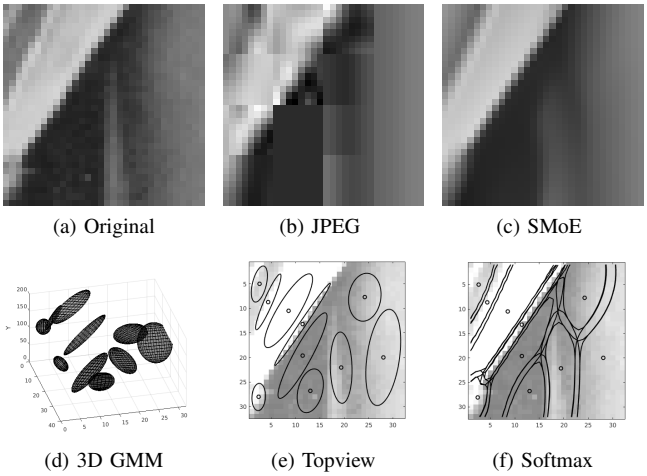
Especially for light field images, SMoE was shown to yield competitive rate-distortion results for low- to mid-level bitrates [8]. For rendering it has three important properties [10]. Firstly, view-rendering is lightweight and pixel-parallel. Secondly, SMoE is a space-continuous representation, thus rendering at arbitrary resolution consists of merely sampling this function. Finally, all local light information in a certain point in the physical space is also localized in the model.

Previous research has shown that real-time rendering of light field images is possible, given appropriate hardware [10]. However, the framerate is heavily dependent on the number of kernels that are involved for the calculation of a single pixel. In this paper, we present an extension of the implementation towards light field video. Secondly, we propose a method to ensure hard real-time constraints by limiting the number of kernels per pixel to evaluate.

II. Steered Mixture-of-Experts

A. Introduction

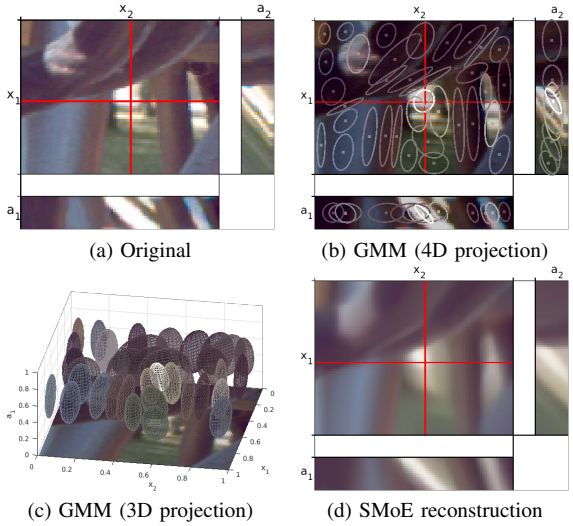
Steered Mixture-of-Experts (SMoE) is a novel framework for the approximation of image modalities with many applications, such as image modality coding, scale conversion (e.g. frame interpolation), and image description (e.g. depth estimation) [6]–[8]. Due to the sparse structure in SMoE, it is readily extendable towards higher dimensional image modalities, such as 6DoF content. This is in stark contrast to traditional image coding schemes which rely on dense sample-grid structures. Moreover, it departs significantly from the conventional coding methods by operating in the spatial domain and thus not using any kind of transform coding. Instead of storing exactly the samples or the transform coefficients that define the image, this method relies on modeling the underlying generative function that could have given rise to the samples.



(a) Original (b) JPEG (c) SMoE

(d) 3D GMM (e) Topview (f) Softmax

Fig. 1. An example of the modeling with 10 components and reconstruction of a 32x32 pixel crop from *Lena* (1a). For a grayscale image, the coordinate space X is 2D and the colorspace Y is 1D. Modeling the joint probability function of both X and Y using a GMM results in 3D Gaussian kernels (1d). Each kernel thus defines a 2D gradient as the *expert* function ($X \mapsto Y$). The gating function is defined by the soft-segmentation (1f). Both JPEG (1b) and SMoE (1c) are coded at 0.35 bpp [6].



(a) Original (b) GMM (4D projection)

(c) GMM (3D projection) (d) SMoE reconstruction

Fig. 2. SMoE applied on a spatial crop of a static 4D light field (*I01 Bikes* [12]), shown in an *epipolar* (EPI) representation in 2a. A GMM of the coordinates (4D) and color amplitudes (3D) is fit as shown in 2b using 35 kernels. A 3D reduction retaining the two spatial dimensions with only one angular dimension is shown in 2c. Finally, the regression based on the GMM is visible in 2d. Note how the white background is approximated by a single kernel.

The function approximation of the underlying generative function is done by identifying coherent, stationary regions in the image modality. Each segment is modeled using a single N-dimensional entity, which we call a *kernel* or *component*. SMoE is based on the divide-and-conquer principle that is present in all *Mixture-of-Experts* (MoE) approaches [11]. The input space is divided in soft-segments using a gating function. Local regressors (or *experts*) are sought that locally approximate the function optimally. The gating function then lets experts collaborate in segments where they are trustworthy.

SMoE is based on the Bayesian, or "alternative" definition of the MoE model [11]. The Bayesian MoE approach models the joint probability of the input space X and the output space Y using a *Gaussian Mixture Model* (GMM). Each Gaussian kernel then simultaneously defines the gating function (soft-segmentation of X) and the local regressors (through the conditional probability function $Y|X$).

In SMoE, where the input space is the *coordinate space* (i.e. sample locations) and the output space is the *color space* (i.e. sample amplitudes), one such Gaussian then corresponds to one kernel as mentioned above. The gating function is thus defined by the probability of a coordinate to belong to a Gaussian, and each Gaussian simultaneously defines an expert function, namely the conditional color amplitudes, given a coordinate. In general, SMoE allows to query the model at any sub-pixel coordinate to yield the most optimal amplitude in a Bayesian sense.

SMoE thus arrives at a sparse representation. The whole image modality is represented as a set of Gaussian kernels. These kernels are defined by their centers and their steering parameters. The coordinate space is 2D, 3D, or 4D in the case of respectively images, video, and static light fields [6]–[8], and analogously 5D for light field video. The color space for color images is conventionally represented as a 3D space, e.g. YCbCr. As the GMM models the joint probability of the coordinate and color space, we thus arrive at 5D, 6D, 7D, and here 8D Gaussian kernels. The parameters of these kernels are typically estimated using computational efficient variations of the *Expectation-Maximization* (EM) algorithm [13]. Due to this likelihood optimization, kernels will steer along the dimensions of the highest correlation, e.g., along spatial or temporal consistencies.

Fig. 1 shows an example of the compression capability of the SMoE approach for coding a 32x32 pixel crop of Lena at 0.35 bits/sample in comparison to JPEG at same rate. Clearly, the edges are reconstructed with convincing quality and sharpness, using merely 10 components [6]. Fig. 2 illustrates SMoE applied to static 4D light fields [8].

B. Theory

The goal of regression is to optimally predict a dependent random vector $Y \in \mathbb{R}^q$ from a known random vector $X \in \mathbb{R}^p$. In SMoE, X corresponds to pixel coordinates (i.e., the 5D coordinate space) and Y to the pixel amplitudes (i.e., the 3D color space). The joint probability function of the coordinate space X and color space Y is modeled as a multi-modal, multi-variate GMM. Each Gaussian kernel then defines a soft-segment in X and a local regressor ($X \mapsto Y$). The local regressor is defined by the mean of the conditional pdf $\mathbf{E}[Y|X = \boldsymbol{x}]$ [6].

Let us assume $D = \{\boldsymbol{x}^i, \boldsymbol{y}^i\}_{i=1}^N$ to be N pixels to be modeled with coordinates $\boldsymbol{x}$ and amplitudes $\boldsymbol{y}$:

$$p_{XY}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}_j, R_j) \tag{1}$$

$$\text{and } \sum_{j=1}^{K} \pi_j = 1, \boldsymbol{\mu}_j = \begin{bmatrix} \boldsymbol{\mu}_{X_j} \\ \boldsymbol{\mu}_{Y_j} \end{bmatrix}, R_j = \begin{bmatrix} R_{X_jX_j} & R_{X_jY_j} \\ R_{Y_jX_j} & R_{Y_jY_j} \end{bmatrix}$$

The parameters of this mixture model with K Gaussian distributions are $\Theta = [\theta_1, \cdots, \theta_K]$, with $\theta_j = (\pi_j, \boldsymbol{\mu}_j, R_j)$, being the population densities, centers, and covariances respectively.

The conditional pdf of the mixture model $Y|X$ is used to derive the regression function [14], [15]. The regression of the model is defined as the expected value $\boldsymbol{y}$ given a sample location $\boldsymbol{x}$ through the conditional. The resulting regression function $m(x)$ is as follows:

$$\hat{\boldsymbol{y}} = m(\boldsymbol{x}) = \mathbf{E}[Y|X = \boldsymbol{x}] = \sum_{j=1}^{K} w_j(\boldsymbol{x}) m_j(\boldsymbol{x}) \tag{2}$$

with mixing weights $w_j(x)$ and regressors $m_j(x)$:

$$w_j(\boldsymbol{x}) = \frac{\pi_j \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{X_j}, R_{X_j X_j})}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{X_i}, R_{X_i X_i})} \tag{3}$$

$$m_j(\boldsymbol{x}) = \boldsymbol{\mu}_{Y_j} + R_{Y_j X_j} R^{-1}_{X_j X_j} (\boldsymbol{x} - \boldsymbol{\mu}_{X_j}), \tag{4}$$

A signal at location x can be predicted by the weighted sum over all K mixture components (Eq. 2). Every mode in the mixture model is considered as an expert and the experts collaborate towards the definition of the regression function.

For the case of light field video [9], $\boldsymbol{x}$ is a 5D vector ($p = 5$) with time dimension t, two angular dimensions (a_1, a_2), and two spatial dimensions (d_1, d_2).

III. Pixel-level Rendering Architecture

Given a set of Gaussian parameters, the goal of this architecture is to reconstruct 2D views which are planar slices of the 5D coordinate space. This architecture consists of two levels of parallelization, similar to [10]. The first level is on block-level, the second level is performed on pixel-level. We further introduce the proposed hard real-time method. The presence of the term "kernels" in SMoE, as well as in GPU architectures creates confusion. In order to avoid this we will refer to "computing kernels" in the case of GPU computing kernels, and "components" when talking about Gaussian steering kernels.

A. Block-level parallelism

A 5D sample location (t, a_1, a_2, d_1, d_2) is reconstructed only from the components that have a significant influence on that location. Due to the competitive nature of training, the reconstruction for one location tends to be reduced to a weighted sum of a very limited set of components. As such the memory requirements are constrained and no unnecessary evaluations are performed.

Based on this observation, we perform a computationally cheap and crude subdivision of the 2D reconstruction plane and the Gaussian components. This subdivision also provides data locality, which is necessary for efficient GPU throughput. The coordinate space is divided into spatial 2D blocks on that reconstruction plane at a specific (t, a_1, a_2), possibly at sub-pixel accuracy. A 5D relevance window larger than the block is used to determine which components are relevant based on their center $\boldsymbol{\mu}_X$. Consequently, each sample inside of this block is reconstructed by that set of relevant components that lay in the vicinity of the reconstruction plane in X. Fig. 3 illustrates how the coordinate space is divided into blocks and dispatched to separate computing kernel functions. Finally, these blocks are processed in parallel.

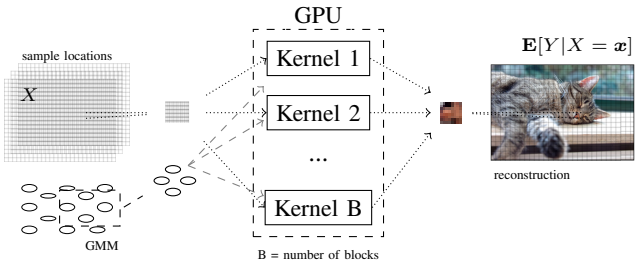


Fig. 3. Block-level parallelism [10]: Each computing kernel is responsible for a block of coordinates. Each block receives a set of Gaussian components that are relevant for this block, which are found by defining a relevance window around this block.

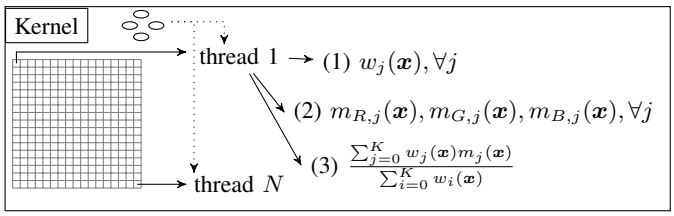


Fig. 4. Pixel-level parallelism [10]: Inside each computing kernel, the kernel dispatches for each sample to be regressed one thread. In each thread a single weight $w_j(x)$ is calculated and three regressors $m_j(x)$ for each color channel. Consequently, in the same thread, the weights are normalized and the weighted sum over each relevant Gaussian component j is calculated. Note that each thread has access to the same set of relevant components and computes the weighted sum over all these components.

B. Pixel-level parallelism

As illustrated in Fig. 4, each pixel within a block is reconstructed independently using Eq. 2. As such, for every block calculated in parallel, pixels are reconstructed simultaneously and full parallelization can be claimed. To achieve this, every pixel/block computation will be mapped efficiently on thread blocks which can then be scheduled to run in parallel.

C. Hard real-time

In [10], we have shown that the rendering speed is mainly dependent on the amount of Gaussian components that one pixel needs to evaluate. Here we propose fixing the number of components that one thread is allowed to evaluate (K_{max}), which thus would allow to define a guaranteed throughput. In order to ensure that the pixel is still reconstructed by the most significant kernels, we choose to rank the kernels based on their priors π_j. These priors indicate the number of original samples this component was responsible for. In case of hard real-time constraints, we want to prioritize larger Gaussian components, e.g. an area in the background that is present in all views for a considerable amount of frames.

IV. Implementation

In this section, we discuss the most important implementation details and differences compared to the version presented

(a) $K_{\max} = 44(474\text{fps})$ (b) $K_{\max} = 355(66\text{fps})$ (c) $K_{\max} = 1423(43\text{fps})$

Fig. 5. Illustration of the visual degradation of *train2*.

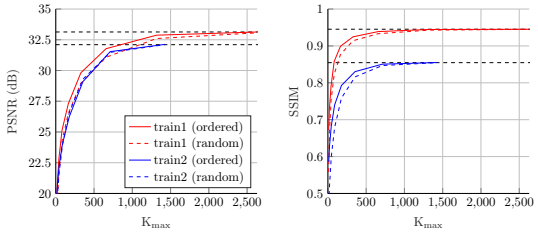


Fig. 6. PSNR and SSIM results for various $K_{\max}$. The dashed lines indicate the PSNR and SSIM values when no component truncation is performed [9].

in [10]. The Khronos' OpenCL API was chosen as it offers both coarse and fine-grained hardware parallelism, has comparable performance to other compute libraries such as NVidia's CUDA, and has extensive portability properties [16], [17].

The formulas for $w_j(\boldsymbol{x})$ and $m_j(\boldsymbol{x})$ in Sec. II require some numerically challenging calculations, e.g. the determinant or matrix inverses. The computation of the determinant and the inverse of large nxn matrices using Leibniz method is inefficient (using $n!$ multiplications) and numerically unstable. This is true even for 5x5 matrices in single precision floating point. We use a closed form implementation of the Choleski factorization using native intrinsic GPU instructions. The dimensionality of our model requires 39 unique floating point values per Gaussian kernel, as opposed to 15 for the 2D and 30 values for the 4D data sets in our previous work [10]. The values are again stored in shared memory to increase the throughput during calculation for all pixels (GPU threads) of a single 16x16x97 data block. Each data block is further segmented on the GPU in 8x8 GPU workgroup blocks.

V. Experimental Evaluation

A. Setup

The machine used has a Intel Core i7 CPU 3720QM @2.6 Ghz with 64GB RAM. The GPU is a four-way SLI NVidia Geforce GTX TITAN X setup with Maxwell architecture. The dataset is based on two SMoE models with $K = 30568$ and $K = 31496$ trained on the light field videos *train1* and *train2* [9], [18], with a dimension cardinality of $(t, a_1, a_2, d_1, d_2) = (84, 8, 8, 352, 512)$ and $(97, 8, 8, 320, 544)$.

The first block-level parallelism (Fig. 3) is performed to build a dataset for evaluating the pixel-level parallelism step in the next subsection. The SMoE model is subdivided in sets of kernels only based on spatial dimensions (d_1, d_2) using blocks of 16x16 and a corresponding relevance window of 36x36. Each set thus contains kernels in the whole (t, a_1, a_2)

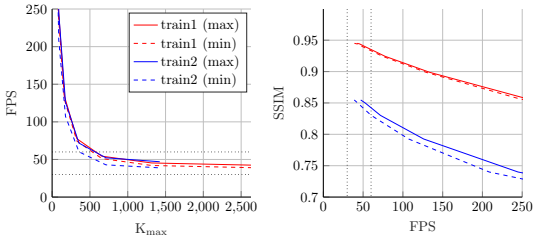


Fig. 7. Left: $K_{\max}$ vs frame-per-second (FPS). Right: Corresponding SSIM values for measured minimum and maximum FPS. The dotted lines mark the 30fps and 60fps borders.

domain. These are transferred to the GPU upfront, and can be used to reconstruct the view for each angle and point in time.

We perform multiple compute passes of 512x512 compute items to reconstruct a complete frame if the frame is larger than 512x512. We report aggregated timings for each video frame, measured 100 times and with a warm-up time of 50 frames to exclude GPU throttling effects.

B. Frame-rate vs. reconstruction quality

In order to evaluate the effectiveness of prioritizing the large components, we compare a randomized and a sorted list of components per block which we then truncate to a predetermined $K_{\max}$. We start with $K_{\max} = \max(K_i)$, K_i being the number of relevant blocks in block i. We then continue dividing the $K_{\max}$ by a factor two.

Fig. 6 illustrates the loss in PSNR and SSIM for a given $K_{\max}$ [19]. Notice an increased quality if the local component selection was based on the priors π_j compared to if the selection was done at random. Fig. 7 illustrates the corresponding achieved minimum and maximum frames-per-second (FPS). When plotting the measured quality vs. the measured timings, we observe a roughly negative linear relation between SSIM and FPS. We can conclude that component truncation is a valid method to obtain a desired frame-rate without a major punishment in quality.

VI. Conclusions and Future Work

In this paper we presented a GPU implementation to render light field videos that are modeled by Steered Mixture-of-Experts. We have shown that we can enforce hard real-time constraints as the rendering speed is dependent on the number of Gaussian components that are being evaluated per pixel. Setting a maximum number of components, while prioritizing large components enforces a fixed arbitrary framerate, albeit with varying reconstruction quality.

In terms of GPU implementation in the future, we plan to apply batched computation of Choleski factorization such as suggested in [20]. As such, we take more advantage of shared memory. Furthermore, this paper proves that there is a lot of potential for other component selection heuristics on the GPU, i.e. using a secondary sliding-window relevance selection along (t, a_1, a_2) dimensions on thread-level, instead of selecting based on π_j.

References

[1] A. T. Hinds, D. Doyen, and P. Carballeira, "Toward the realization of six degrees-of-freedom with compressed light fields," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2017, pp. 1171–1176.

[2] I. Ihrke, J. Restrepo, and L. Mignard-Debise, "Principles of light field imaging: briefly revisiting 25 years of research," *IEEE Signal Processing Magazine*, vol. 33, no. 5, pp. 59–69, 2016.

[3] E. Adelson and J. Bergen, *The plenoptic function and the elements of early vision*. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991.

[4] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, New York, New York, USA: ACM Press, 1996, pp. 31–42.

[5] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu, "Light field image processing: an overview," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–1, 2017.

[6] R. Verhack, T. Sikora, L. Lange, G. Van Wallendael, and P. Lambert, "A universal image coding approach using sparse steered mixture-of-experts regression," in *IEEE Proc. Int. Conf. on Image Processing (ICIP)*, IEEE, 2016, pp. 2142–2146.

[7] L. Lange, R. Verhack, and T. Sikora, "Video representation and coding using a sparse steered mixture-of-experts network," in *Picture Coding Symposium (PCS)*, 2016.

[8] R. Verhack, T. Sikora, L. Lange, R. Jongebloed, G. Van Wallendael, and P. Lambert, "Steered mixture-of-experts for light field coding, depth estimation, and processing," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, Ed., IEEE, 2017, pp. 1183–1188.

[9] R. Verhack, G. Van Wallendael, M. Courteaux, P. Lambert, and T. Sikora, "Progressive modeling of steered mixture-of-experts for light field video approximation," in *Submitted to Picture Coding Symposium '18*, 2018.

[10] V. Avramelos, R. Verhack, I. Saenen, G. Van Wallendael, B. Goossens, and P. Lambert, "Highly parallel steered mixture-of-experts rendering at pixel-level for image and light field data," *Submitted to Journal on Real-Time Image Processing*, 2018.

[11] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 23, no. 8, pp. 1177–1193, 2012.

[12] M. Rerabek and T. Ebrahimi, "New light field image dataset," in *8th International Conference on Quality of Multimedia Experience (QoMEX)*, 2016.

[13] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[14] H Sung, "Gaussian mixture regression and classification," PhD thesis, Rice University, 2004.

[15] G. Bugmann, "Normalized gaussian radial basis function networks," *Neurocomputing*, vol. 20, no. 1-3, pp. 97–110, 1998.

[16] Khronos OpenCL Working Group, "The opencl specification. version 2.0. revision 29," Tech. Rep., 2015.

[17] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, p. 40, 2008.

[18] T.-C. Wang, J.-Y. Zhu, N. K. Kalantari, A. A. Efros, and R. Ramamoorthi, "Light field video capture using a learning-based hybrid imaging system," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–13, 2017.

[19] Z Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[20] F. Lemaitre and L. Lacassagne, "Batched cholesky factorization for tiny matrices," in *2016 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, IEEE, 2016, pp. 130–137.